Objective:

To train two agents to control rackets to bounce a ball over a net. If an agent hits the ball over the net, it receives a reward of +0.1. If an agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01.

Environment:

The observation space consists of 8 variables corresponding to the position and velocity of the ball and racket. Each agent receives its own, local observation. Two continuous actions are available, corresponding to the movement toward (or away from) the net, and jumping.

Implementation:

1. The algorithm being used is from [this paper](*https://arxiv.org/pdf/1509.02971.pdf*), _Continuous Control with Deep Reinforcement Learning by researchers at Google Deepmind.
2. We have 2 agents which use their separate networks to learn.
3. Both the agents share the same replay buffer memory.
4. Batch Normalisation has helped to achieve the results faster.
5. Used the Ornstein-Uhlenbeck process, as suggested by [Google DeepMind](*https://arxiv.org/pdf/1509.02971.pdf*)

Model:

We are using DDPG, Actor-Critic algorithm for both the agents. Specifications are as follows:

Actor:

1. Input Layer size: 24
2. Hidden Layers: [128, 256]
3. Batch Normalization
4. Hidden Layer Activation Function: Relu
5. Output Layer size: 4
6. Output Layer Activation Function: tanh

Critic:

1. Input Layer size: 24
2. First Hidden Layer size: 128
3. Batch Normalization
4. Activation function: Relu
5. Concatenation of Actor output and first hidden layer output

6. Second Hidden Layer size: 256
7. Output size: 1

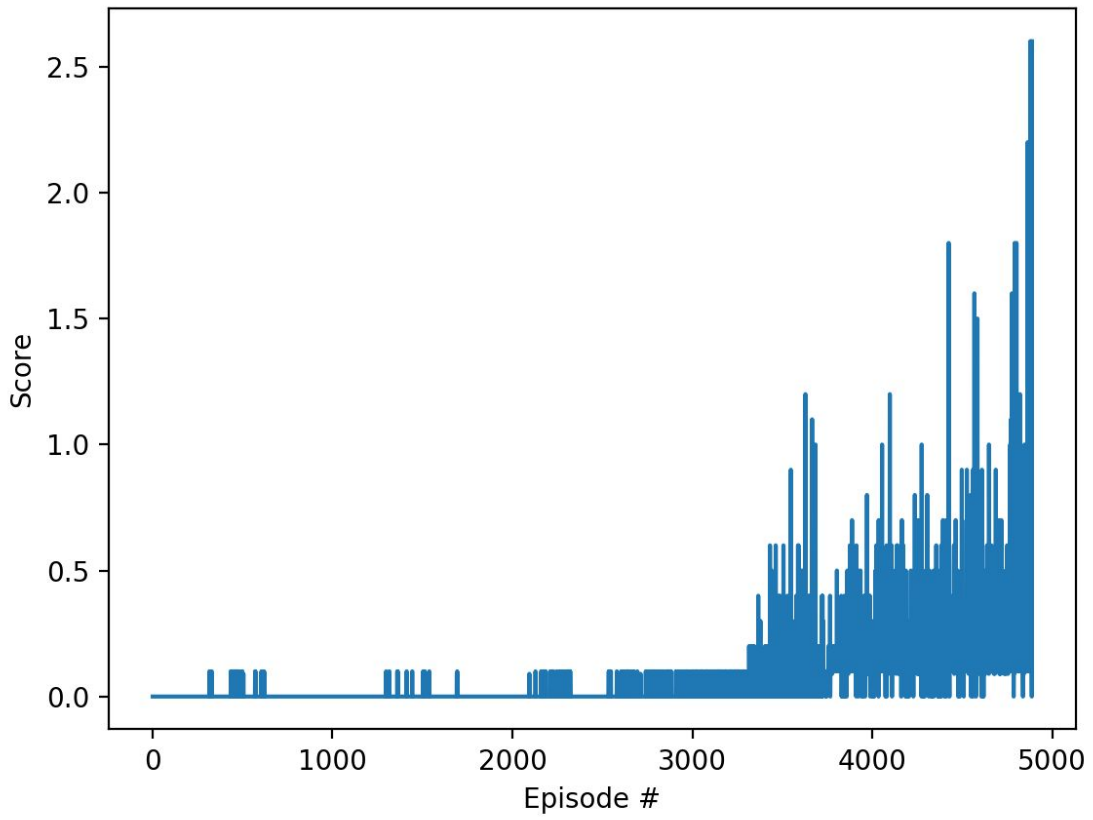
Hyperparameters:

1. Buffer Size:
    a. Description: the size of replay memory
    b. Value: int(1e5)
2. Batch size:
    a. Description: number of samples being used in one iteration
    b. Value: 64
3. Gamma:
    a. Description: discount factor
    b. Value: 0.99
4. Tau:
    a. Description: factor for the soft update of the target model
    b. Value: 1e-3
5. LR:
    a. Description: learning rate
    b. Value: 5e-4
6. Update_every:
    a. Description: After how many samples we need to learn
    b. Value: 4
7. Learn-every:
    a. Description: After how many timesteps we need to learn
    b. Value: 20
8. Learn-num:
    a. Description: how many times do we have have to learn
    b. Value: 10

Output:

1. The environment is solved in 4887 episodes to reach an average reward of 0.5 over 100 consecutive episodes.
2. 2. Weights are being saved at checkpoint_actor1.pth, checkpoint_critic1.pth, checkpoint_actor2.pth, checkpoint_critic2.pth
3. 3. The graph below shows the final results:

The graph below shows the final results:



Score
2.5
2.0
1.5
1.0
0.5
0.0
0
1000
2000
3000
4000
5000
Episode #